Tablice w PHP

W programowaniu często musimy przechowywać więcej niż jedną wartość – jak to robimy w zmiennej. Gdybyśmy chcieli np. zapamiętać listę produktów w sklepie internetowym albo imiona studentów w dzienniku, to zamiast tworzyć mnóstwo pojedynczych zmiennych (

`$produkt1`, `$produkt2`, `$produkt3`…), używamy rzecz jasna tablicy.

Tablica w PHP to zatem tzw. struktura danych (pojemnik na informacje), który pozwala przechowywać wiele kolejnych wartości pod jedną nazwą. W PHP mamy do dyspozycji kilka różnych tablic, w zależności od tego czy numerujemy kolejne szufladki, nazywamy je albo zagnieżdżamy jedną tablicę w drugiej (tworząc "kolekcję" tablic):

- Tablice indeksowane (numeryczne) – klucze są liczbami
- Tablice asocjacyjne – indeksy szufladek zamiast numerów mają przypisane nazwy – asocjacja oznacza skojarzenie.
- Tablice wielowymiarowe – tablice wewnątrz tablic (zagnieżdżone).

Tablica indeksowana (indeks numeryczny)

W tablicy indeksowanej każdy element ma przypisany numer (indeks), zaczynający się oczywiście – jak to w programowaniu – od zera. Zobaczmy podstawowe mechaniki pracy z tablicą:

Tworzenie tablicy:

```php
$owoce = array("jabłko", "banan", "gruszka");
```

Odczyt wartości z tablicy po indeksie szuflady:

```php
echo $owoce[0]; // wypisze: jabłko
```

Zapis nowej wartości (podmiana elementu):

```php
$owoce[1] = "kiwi"; // teraz zamiast banana, pod indeksem 1 jest kiwi
```

Dodanie nowego elementu (na koniec):

```php
$owoce[] = "truskawka"; // automatycznie dostanie indeks 3 (czwarty z kolei)
```

Tablica asocjacyjna (indeks tekstowy, czyli nazwa szuflady)

W tablicy asocjacyjnej nie używamy liczb jako indeksów – zamiast tego każda wartość ma nazwany (czytelny) klucz. Taki typ tablicy jest bardzo przydatny, gdy dane mają jakiś kontekst – np. opis osoby albo kiedy informacje pochodzą z bazy danych (możemy wówczas użyć nazw kolumn takich samych jak w tabeli). Pora na przegląd podstawowych mechanik:

Tworzenie tablicy:

```php
$osoba = array(
  "imie" => "Jan",
  "nazwisko" => "Kowalski",
  "wiek" => 30
);
```

Odczyt wartości po nazwie szuflady:

```php
echo $osoba["imie"]; // wypisze: Jan
```

Zapis nowej wartości (nadpisanie):

```php
$osoba["wiek"] = 31;
```

Dodanie nowej pary “klucz → wartość”:

```php
$osoba["miasto"] = "Warszawa";
```

3. Tablica wielowymiarowa

Tablice wielowymiarowe to tablice zawierające inne tablice – np. lista osób, gdzie każda osoba ma swoje dane zapisane w osobnej tablicy. Możemy to porównać do tabeli – z wierszami i kolumnami (dwa wymiary, czyli inaczej dwie współrzędne dostępu do wartości w tablicy).

Tworzenie tablicy wielowymiarowej:

```php
$uczniowie = array(
  array("imie" => "Anna", "nazwisko" => "Nowak", "wiek" => 17),
  array("imie" => "Tomasz", "nazwisko" => "Lis", "wiek" => 18),
);
```

Odczyt wartości z konkretnej tablicy wewnętrznej:

```php
echo $uczniowie[1]["nazwisko"]; // wypisze: Lis
```

Zapis nowej wartości w zagnieżdżonej tablicy:

```php
$uczniowie[0]["wiek"] = 18; // zmieniamy wiek pierwszej osoby
```

Pętle do pracy z tablicami w PHP

Najczęściej używaną pętlą do przechodzenia po elementach tablicy w PHP jest

`foreach` – działa niezależnie od indeksów numerycznych i automatycznie pobiera kolejne elementy. Ale oczywiście możemy też użyć klasycznych pętli `for` lub `while`, szczególnie gdy zależy nam na pracy z indeksami liczbowymi.

`foreach` – przechodzenie przez wszystkie elementy tablicy, dostęp do wartości:

```php
$owoce = array("jabłko", "banan", "gruszka");
foreach ($owoce as $owoc) {
    echo $owoc . "<br>";
}
// jabłko
// banan
// gruszka
```

`foreach` z dostępem do klucza i wartości:

```php
$osoba = array("imie" => "Anna", "wiek" => 25);
foreach ($osoba as $klucz => $wartosc) {
```

```
    echo $klucz . ": " . $wartosc . "<br>";
}
// imie: Anna
// wiek: 25
```

`for` – działa dobrze z tablicami indeksowanymi numerycznie:

```
$liczby = array(10, 20, 30, 40);
for ($i = 0; $i < count($liczby); $i++) {
    echo $liczby[$i] . "<br>";
}
// 10
// 20
// 30
// 40
```

`while` – przykład z licznikiem i tablicą:

```
$kolory = array("czerwony", "zielony", "niebieski");
$i = 0;
while ($i < count($kolory)) {
    echo $kolory[$i] . "<br>";
    $i++;
}
// czerwony
// zielony
// niebieski
```

Przegląd użytecznych funkcji tablicowych w PHP

W PHP istnieje prawdziwe bogactwo funkcji współpracujących z tablicami. Na początku tak długa lista może nas wręcz przerażać, lecz pamiętajmy że każda taka poznana przez nas metoda może nam zaoszczędzić sporo czasu na egzaminie.

`count()` – zwraca liczbę elementów w tablicy.

```
$owoce = array("jabłko", "banan", "gruszka");
echo count($owoce); // 3
```

`array_push()` – dodaje jeden lub więcej elementów na koniec tablicy.

```
$liczby = array(1, 2, 3);
array_push($liczby, 4, 5);
print_r($liczby); // Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 )
```

`array_pop()` – usuwa ostatni element z tablicy i go zwraca.

```
$kolory = array("czerwony", "zielony", "niebieski");
$usuniety = array_pop($kolory);
echo $usuniety; // niebieski
```

`array_shift()` – usuwa pierwszy element z tablicy i przesuwa pozostałe w lewo.

```php
$kolejka = array("Anna", "Tomek", "Ola");
$pierwszy = array_shift($kolejka);
echo $pierwszy; // Anna
```

`in_array()` – sprawdzi, czy dana wartość znajduje się w tablicy.

```php
$wartosci = array("a", "b", "c");
if (in_array("b", $wartosci)) {
    echo "Znaleziono"; // Znaleziono
}
```

`array_key_exists()` – sprawdza, czy dany klucz istnieje w tablicy asocjacyjnej.

```php
$osoba = array("imie" => "Jan", "wiek" => 30);
if (array_key_exists("wiek", $osoba)) {
    echo "Klucz istnieje"; // Klucz istnieje
}
```

`array_merge()` – łączy dwie lub więcej tablic w jedną nową.

```php
$a = array("a", "b");
$b = array("c", "d");
$polaczona = array_merge($a, $b);
print_r($polaczona); // Array ( [0] => a [1] => b [2] => c [3] => d )
```

`sort()` – sortuje tablicę rosnąco według wartości, modyfikując ją w miejscu.

```php
$liczby = array(3, 1, 4);
sort($liczby);
print_r($liczby); // Array ( [0] => 1 [1] => 3 [2] => 4 )
```

`array_reverse()` – odwraca kolejność elementów w tablicy i zwraca nową tablicę.

```php
$oryginalna = array(1, 2, 3);
$odwrocona = array_reverse($oryginalna);
print_r($odwrocona); // Array ( [0] => 3 [1] => 2 [2] => 1 )
```

`array_slice()` – wycina fragment tablicy, zwracając nową tablicę bez zmieniania oryginału.

```php
$owoce = array("jabłko", "banan", "gruszka", "śliwka");
$kawalek = array_slice($owoce, 1, 2);
print_r($kawalek); // Array ( [0] => banan [1] => gruszka )
```

`array_unique()` – usuwa z tablicy duplikaty i zwraca nową tablicę.

```php
$liczby = array(1, 2, 2, 3, 3, 3);
$unikalne = array_unique($liczby);
print_r($unikalne); // Array ( [0] => 1 [1] => 2 [3] => 3 )
```

`array_keys()` – zwraca wszystkie klucze tablicy jako nową tablicę.

```php
$osoba = array("imie" => "Anna", "wiek" => 25);
$klucze = array_keys($osoba);
print_r($klucze); // Array ( [0] => imie [1] => wiek )
```

`array_values()` – zwraca wszystkie wartości tablicy jako nową tablicę, bez kluczy.

```php
$osoba = array("imie" => "Anna", "wiek" => 25);
$wartosci = array_values($osoba);
print_r($wartosci); // Array ( [0] => Anna [1] => 25 )
```

`implode()` – łączy elementy tablicy w jeden łańcuch znaków, używając separatora.

```php
$slowa = array("PHP", "jest", "super");
$zdanie = implode(" ", $slowa);
echo $zdanie; // PHP jest super
```

`explode()` – dzieli łańcuch znaków na tablicę według separatora.

```php
$tekst = "jabłko,banan,gruszka";
$tablica = explode(",", $tekst);
print_r($tablica); // Array ( [0] => jabłko [1] => banan [2] => gruszka )
```

`is_array()` – sprawdza, czy dana zmienna jest tablicą.

```php
$zmienna = array(1, 2, 3);
if (is_array($zmienna)) {
    echo "To jest tablica"; // To jest tablica
}
```

`in_array()` – sprawdza, czy podana wartość istnieje w tablicy – zwraca true lub false.

```php
$owoce = array("jabłko", "banan", "gruszka");
if (in_array("banan", $owoce)) {
    echo "Banan znajduje się w tablicy"; // Banan znajduje się w tablicy
}
```

`array_search()` – zwraca indeks (lub klucz) pierwszego wystąpienia danej wartości w tablicy.

```php
$kolory = array("czerwony", "zielony", "niebieski");
$pozycja = array_search("zielony", $kolory);
echo $pozycja; // 1
```

`array_merge()` – łączy dwie lub więcej tablic w jedną.

```php
$a = array(1, 2);
$b = array(3, 4);
$c = array_merge($a, $b);
```

```
print_r($c); // Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )
```

`array_diff()` – porównuje tablice i zwraca elementy obecne w pierwszej tablicy, ale nieobecne w pozostałych.

```
$a = array("a", "b", "c", "d");
$b = array("a", "c");
$roznica = array_diff($a, $b);
print_r($roznica); // Array ( [1] => b [3] => d )
```

`array_key_exists()` – sprawdza, czy dany klucz istnieje w tablicy asocjacyjnej.

```
$osoba = array("imie" => "Jan", "wiek" => 30);
if (array_key_exists("wiek", $osoba)) {
    echo "Klucz 'wiek' istnieje"; // Klucz 'wiek' istnieje
}
```

Po tak solidnej powtórce fundamentalnych mechanik oraz po poznaniu tak wielkiego wachlarza metod, praca z tą strukturą danych powinna być w PHP wręcz przyjemnością. Zachęcam do wielu ćwiczeń z nimi, zwłaszcza podczas realizowania zadań wymagających od nas fetchowania informacji z bazy danych.